TREE STRUCTURE

Hua Huang

FIELD OF INVENTION

[0001] This invention relates to a tree structure for storing data.

DESCRIPTION OF RELATED ART

[0002] Fig. 1 illustrates a conventional N-tree structure 10 for storing an N-dimensional object. Root 12 of tree structure 10 corresponds to the entire object. If the object is completely uniform (i.e., all the data are the same), then its data are stored in root 12 and root 12 becomes the entire tree. Otherwise, a new level is added to tree structure 10 with nodes 14-0, 14-1 … 14-N that correspond to subdivisions of the object. If a subdivision is completely uniform, such as node 14-N, then its data are stored in its node and the node becomes a leaf of tree structure 12. A non-uniform subdivision adds a new level to tree structure 12, with child nodes 16-0, 16-1 … 16-N that correspond to sub-subdivisions of the non-uniform subdivision. This structure continues until the smallest unit of the object is reached so the object cannot be further divided.

[0003] As Fig. 1 illustrates, each node must have N pointers to its child node. If N is a large number, then tree structure 10 consume a large amount of memory just to store the pointers. Thus, what is needed is a more efficient tree structure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] Fig. 1 illustrates a conventional N-tree structure.

[0005] Fig. 2 illustrates a tree structure in one embodiment of the invention.

[0006] Fig. 3 illustrates a node of the tree structure of Fig. 2 in one embodiment of the invention.

[0007] Fig. 4 illustrates a method to search the tree structure of Fig. 2 in one embodiment of the invention.

[0008] Use of the same reference numbers in different figures indicates similar or identical elements.

SUMMARY

[0009] In one embodiment of the invention, a data structure includes a parent node and a plurality of children nodes of the parent node arranged in an order. The parent node includes a first pointer to one of the children nodes and a second pointer to another of the children nodes. Each child node includes a third pointer to a next child node in the order and a fourth pointer to a previous child node in the order. The parent node can further include a fifth pointer to a child node that was last queried.

DETAILED DESCRIPTION

[0010] Fig. 2 illustrates a data structure 50 for storing an object, such as an image, in one embodiment of the invention. Data structure 50 starts with a root node 52 that represents the entire object. Root node 52 is linked by pointers pHead, pTail, and pCursor to the next level of nodes 54-0, 54-1, 54-2 ... 54-N. Nodes 54-0 to 54-N correspond to subdivisions of the object in a predetermined order. In one embodiment, pointer pHead points from a parent node, such as root node 52, to a specific child node in the predetermined order, such as the first child node 54-0. In one embodiment, pointer pTail points from a parent node, such as root node 52, to another specific child node in the predetermined order, such as the last child node 54-N. In one embodiment, pointer pCursor points from a parent node, such as root node 52, to a child node that was last queried in a search.

[0011] Each of nodes 54-0 to 54-N is linked by pointers pNext and pPreview to two other nodes in the same level. In one embodiment, pointer pNext points from a node, such as node 54-1, to the next node in the predetermined order, such as node 54-2. In one embodiment, pointer pPreview points from a node, such as node 54-1, to the previous node in the predetermined order, such as node 54-0.

[0012] The above described structure 56 is the basic substructure of data structure 50 and can be repeated to represent the object in ever finer detail. For example, each of nodes 54-0 to 54-N can be linked by pointers pHead, pTail, and pCursor to the next level of children nodes 58-0, 58-1, 58-2 ... 58-N as described above. Each of nodes 58-0 to 58-N can be linked by pointers pNext and pPreview to two other nodes in the same level.

[0013] Fig. 3 illustrates the pointers stored in a node in data structure 50. Specifically, each node stores pointers pHead, pTail, pNext, pPreview, and pCursor, which are used to search for nodes as described above and hereafter.

[0014] Fig. 4 is a flowchart of a method 80 for searching a requested node in data structure 50 (Fig. 2) in one embodiment of the invention.

[0015] In step 82, the lineage of the requested node is determined. For example, if the requested node is node 58-2, then the lineage of node 58-2 includes a parent node 54-2 and a grandparent node 52. The lineage of the requested node can be determined by the predetermined order in which the object is divided into nodes.

[0016] In step 84, root node 52 of data structure 50 (Fig. 2) is selected.

[0017] In step 86, the shortest path from the selected node through the next level to the requested node is determined. Specifically, the paths from pointers pHead, pTail, and pCursor of the selected node are compared to determine the shortest path to the requested node if it is located in the next level, or one of the ancestor nodes of the requested node if the requested node is not located in the next level.

[0018] In most applications, nodes are requested in order (forward or backward). For example, nodes 54-0 to 54-N are requested in order. Thus, the quickest path to the requested node results from following pointer pCursor of root node 52 to the last requested node, and then following pointer pNext or pPreview to the currently requested node. In another example, nodes 58-0 to 58-N are requested in order. Thus, the quickest path to the requested node most often results in following pointer pCursor of root node 52 to node 54-2, then following pointer pCursor of node 54-2 to the last requested node, and then following pointer pNext or pPreview to the currently requested node. Thus, the quickest path most often results from following pointers pCursor to the level of the currently requested node, and then following pointer pNext or pPreview to the currently requested node.

[0019] In step 88, pointer pCursor for the selected node is updated to point to the requested node or its ancestor in the next level.

[0020] In step 90, it is determined if the requested node or its ancestor node has been found in the next level. If the requested node has been found, then step 90 is followed by step 94. If its ancestor node has been found, then step 90 is followed by step 92.

[0021] In step 92, the ancestor node is selected. Step 92 is followed by step 86 and again the shortest path from the selected node through the next level to the requested node is determined. These steps are repeated until the requested node has been found.

[0022] In step 94, the requested node is returned in response to the request and ends method 80.

[0023] Various other adaptations and combinations of features of the embodiments disclosed are within the scope of the invention. Numerous embodiments are encompassed by the following claims.